

INTRODUCTION

The goal of this project is to leverage the Snowflake data warehouse in conjunction with the Data Build Tool (dbt) to implement Slowly Changing Dimensions (SCD) Types. This effectively handle changing customer data while guaranteeing the correctness of current data and the preservation of historical records. Logstash can be used to ingest CSV files straight into Snowflake tables. The system was configured with Snowflake as the cloud data warehouse and dbt as the transformation and modeling tool. The project's main focus is incremental loading, in which DBT models are made to handle only fresh or updated data, improving performance and cutting down on pointless calculation.

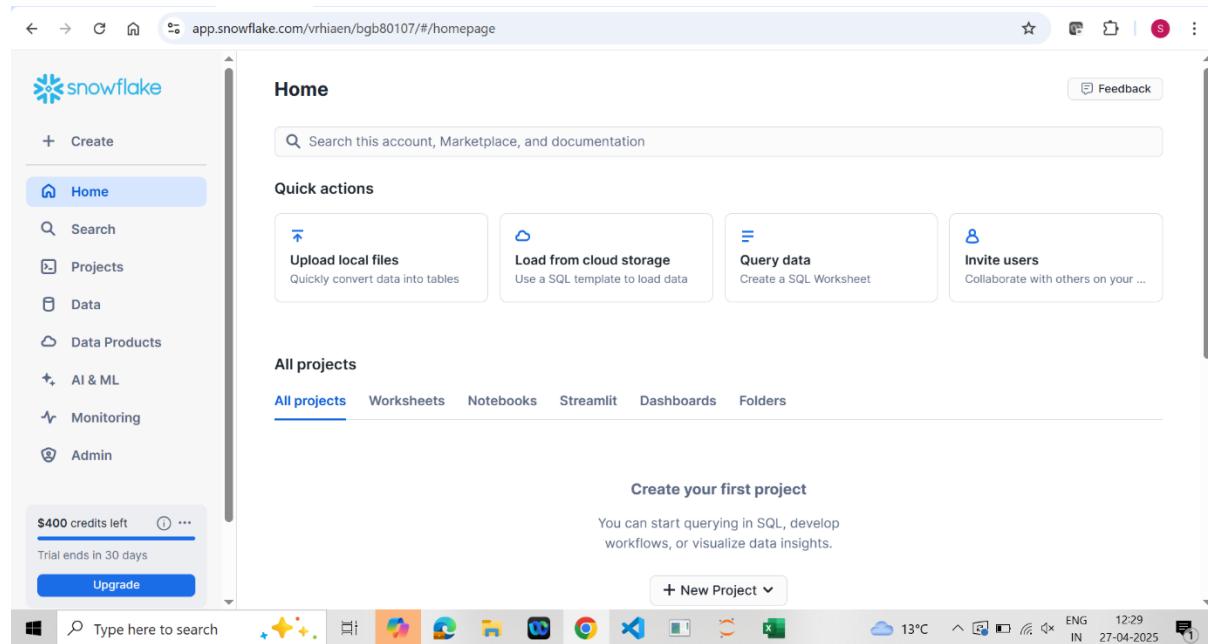
Different business requirements are reflected in the implementation of each form of SCD. When historical tracking is not required, SCD1 offers a simple solution by updating existing entries without keeping any history. By designating old records as inactive and adding new records with updated values, SCD2 maintains the entire history and provides a thorough audit trail of changes over time. By adding new columns, like prev_email, to capture only the most recent change, SCD3 is used to maintain a limited amount of history within a single record. Lastly, by keeping a current table that is frequently updated and a separate history table that records all previous modifications, SCD4 integrates the functionality of SCD1 and SCD2.

To guarantee data quality, the project contains extensive testing and validation using both custom and built-in DBT tests. These include valid flag checks, null value verification, and, when necessary, uniqueness enforcement. To increase performance and maintainability, optimization strategies including incremental model design and modular SQL scripting have also been used. Robust error management and logging are integrated throughout the workflow, guaranteeing dependable data processing and enabling transparent tracking of data concerns.

Environment setup and repository configuration

Snowflake setup

Snowflake account is created as shown in the below screenshot.



Creation of Database

Database called 'CUSTOMER_DATA' in snowflake is created by using SQL queries.

CREATE OR REPLACE DATABASE CUSTOMER_DATA;

The screenshot shows the Snowflake UI interface. On the left, there's a sidebar with icons for databases, worksheets, and other management tasks. The main area shows a timeline from 2025-04-27 4:06pm to 2025-04-27 4:13pm. A query window is open with the following SQL code:

```
1 -- Step 1: Create the database first
2 CREATE OR REPLACE DATABASE CUSTOMER_DATA;
3
```

The results pane shows a single row: "Database CUSTOMER_DATA successfully created." Below the results, the "Query Details" section indicates a duration of 147ms and 1 row.

Creation of Schema

Schema called 'CUSTOMER_SCHEMA' is created under 'CUSTOMER_DATA' database by using below queries.

USE DATABASE CUSTOMER_DATA;

CREATE OR REPLACE SCHEMA CUSTOMER_SCHEMA;

The screenshot shows the Snowflake UI interface. The sidebar and timeline are identical to the previous screenshot. A query window is open with the following SQL code:

```
6
7
8 -- Step 3: Now create the schema inside it
9 CREATE OR REPLACE SCHEMA CUSTOMER_SCHEMA;
10
```

The results pane shows a single row: "Schema CUSTOMER_SCHEMA successfully created." Below the results, the "Query Details" section indicates a duration of 48ms and 1 row.

GENERATING CSV WITH FAKER

A csv file called sample_customer_data.csv is generated using faker().

```

File Edit View Insert Cell Kernel Widgets Help
Not Connected Trusted Python 3
File + < > Run Code
def generate_customer_data(num_records):
    data = []
    for i in range(1, num_records + 1):
        customer_id = i
        customer_name = fake.name()
        email = fake.email()
        join_date = fake.date_between(start_date='-5y', end_date='today').strftime('%Y-%m-%d')
        active = random.choice([True, False])
        data.append([customer_id, customer_name, email, join_date, active])
    return data

# Define Desktop Path for Output
desktop_path = os.path.join(os.path.expanduser("~/"), "Desktop")
output_file = os.path.join(desktop_path, 'sample_customer_data.csv')

# Generate data
customer_data = generate_customer_data(100)

# Write to CSV
with open(output_file, mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(['customer_id', 'customer_name', 'email', 'join_date', 'active'])
    writer.writerows(customer_data)

print(f"Sample customer data CSV file created on Desktop: {output_file}")

```

Sample customer data CSV file created on Desktop: C:\Users\Srilakshmi N Murthy\Desktop\sample_customer_data.csv

LOGSTASH INSTALLATION

Logstash is used to ingest data into Snowflake. The below screenshot explains about installing Logstash packages.

```

D:\logstash\logstash-9.0.0\bin\logstash.bat -f logstash.conf
[2025-04-27T21:07:42,561][INFO ][logstash.runner] Logstash shut down.
[2025-04-27T21:07:42,567][FATAL ][org.logstash.Logstash] Logstash stopped processing because of an error: (SystemExit) exit
org.jruby.exceptions.SystemExit: (SystemExit) exit
    at org.jruby.RubyKernel.exit(org/jruby/RubyKernel.java:924) ~[jruby.jar:?]
    at org.jruby.RubyKernel.exit(org/jruby/RubyKernel.java:883) ~[jruby.jar:?]
    at D:\logstash\logstash_minus_9_dot_0_dot_0.lib.bootstrap.environment.<min>(D:\logstash\logstash-9.0.0\lib\bootstrap\environment.rb:90) ~[?:?]

D:\logstash\logstash-9.0.0\bin\logstash.bat -f logstash.conf
"Using bundled JDK: D:\logstash\logstash-9.0.0\jdk\bin\java.exe"
Sending Logstash logs to D:\logstash\logstash-9.0.0\logs which is now configured via log4j2.properties
[2025-04-27T21:10:14,895][INFO ][logstash.runner] Log4j configuration path used is: D:\logstash\logstash-9.0.0\config\log4j2.properties
[2025-04-27T21:10:14,899][WARN ][logstash.runner] The use of JAVA_HOME has been deprecated. Logstash 8.0 and later ignores JAVA_HOME and uses the bundled JDK. Running Logstash with the bundled JDK is recommended. The bundled JDK has been verified to work with each specific version of Logstash, and generally provides best performance and reliability. If you have compelling reasons for using your own JDK (organizational-specific compliance requirements, for example), you can configure LS_JAVA_HOME to use that version instead.
[2025-04-27T21:10:14,899][INFO ][logstash.runner] Starting Logstash {"logstash.version"=>"9.0.0", "jruby.version"=>"jruby 9.4.9.0 (3.1.4) 2024-11-04 547c6b150e OpenJDK 64-Bit Server VM 21.0.6+7-LTS on 21.0.6+7-LTS +indy +jit [x86_64-mswin32]"}
[2025-04-27T21:10:14,902][INFO ][logstash.runner] JVM bootstrap flags: [-Xms1g, -Xmx1g, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djruby.compile.invokedynamic=true, -XX:+HeapDumpOnOutOfMemoryError, -Djava.security.egd=file:/dev/urandom, -Dlog4j2.isThreadContextMapInheritable=true, -Djruby.reexec.interuptible=true, -Djdk.io.File.enabledADS=true, --add-exports=jdk.compiler/com.sun.tools.javac.api=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.tree>All-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.util=ALL-UNNAMED, --add-opens=java.base/java.security=ALL-UNNAMED, --add-opens=java.base/java.io=ALL-UNNAMED, --add-opens=java.base/java.nio.channels=ALL-UNNAMED, --add-opens=java.base/java.nio.channels=ALL-UNNAMED, --Dio.nettyallocator.maxOrder=11]

```

The below screenshot is about logstash.conf file where we give all the credentials of snowflake like account details, database, schema etc for connection.

```

logstash - Notepad
File Edit Format View Help
}

filter {
    cs {
        separator => ","
        columns => ["customer_id", "customer_name", "email", "join_date", "active"]
        skip_header => true # Skips the header row if your CSV has one
    }
    mutate {
        convert => { "customer_id" => "integer" }
        convert => { "active" => "boolean" }
    }
    date {
        match => ["join_date", "yyyy-MM-dd"]
        target => "join_date"
    }
}

output {
    snowflake {
        user => "SrilakshmiN"
        password => "*****"
        account => "o478682"
        warehouse => "COMPUTE_WH"
        database => "CUSTOMER_DATA"
        schema => "CUSTOMER_SCHEMA"
        table => "CUSTOMERS_PARSED"
        autocommit => true
    }
}

```

The below code explains the conversion of auto generated csv file into json format through Logstash.

```

bin\logstash.bat -f logstash.conf
{
    "name" => "LAPTOP-C90V9F1U"
},
    "email" => "heidi43@example.org",
    "log" => {
        "file" => {
            "path" => "C:/Users/Srilakshmi N Murthy/Desktop/sample_customer_data.csv"
        }
    }
{
    "@timestamp" => 2025-04-28T04:17:34.485032800Z,
    "event" => {
        "original" => "47,Mary Franklin,josephlee@example.net,2025-01-15,False\r"
    },
    "message" => "47,Mary Franklin,josephlee@example.net,2025-01-15,False\r",
    "customer_id" => 47,
    "active" => false,
    "join_date" => 2025-01-15T08:00:00.000Z,
    "customer_name" => "Mary Franklin",
    "@version" => "1",
    "host" => {
        "name" => "LAPTOP-C90V9F1U"
    },
    "email" => "josephlee@example.net",
    "log" => {
        "file" => {
            "path" => "C:/Users/Srilakshmi N Murthy/Desktop/sample_customer_data.csv"
        }
    }
{
    "@timestamp" => 2025-04-28T04:17:34.444103Z,
    "event" => {
        "original" => "55,Julia Collier,comptonsara@example.org,2021-02-06,True\r"
    },
    "message" => "55,Julia Collier,comptonsara@example.org,2021-02-06,True\r",
    "customer_id" => 55,
    "active" => true,
    "join_date" => 2021-02-06T08:00:00.000Z,
    "customer_name" => "Julia Collier",
    "@version" => "1",
}

```

Pipeline Running

This is the Logstash pipeline where the data is automatically loaded inside Snowflake environment. The below screenshot is about pipeline logs.

```

D:\logstash\logstash-9.0.0\bin>logstash.bat -f D:\logstash\logstash-9.0.0\logstash.conf
Using bundled JDK: D:\logstash\logstash-9.0.0\jdk\bin\java.exe
Sending Logstash logs to D:/logstash/logstash-9.0.0/logs which is now configured via log4j2.properties
[2025-04-27T21:46:56.987][INFO ][logstash.runner] Log4j configuration path used is: D:\logstash\logstash-9.0.0\config\log4j2.properties
[2025-04-27T21:46:56.912][WARN ][logstash.runner] The use of JAVA_HOME has been deprecated. Logstash 8.0 and later ignores JAVA_HOME and uses the bundled JDK. Running Logstash with the bundled JDK is recommended. The bundled JDK has been verified to work with each specific version of Logstash, and generally provides best performance and reliability. If you have compelling reasons for using your own JDK (organizational-specific compliance requirements, for example), you can configure LS_JAVA_HOME to use that version instead.
[2025-04-27T21:46:56.913][INFO ][logstash.runner] Starting Logstash {"logstash.version"=>"9.0.0", "jruby.version"=>"jruby 9.4.9.0 (3.1.4) 2024-11-04 547cb1b150e OpenJDK 64-Bit Server VM 21.0.6+7-LTS on 21.0.6+7+indy +jit [x86_64-mswin32]"}
[2025-04-27T21:46:56.916][INFO ][logstash.runner] JVM bootstrap flags: [-Xms1g, -Xmx1g, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djruby.compile.invokedynamic=true, -XX:+HeapDumpOnOutOfMemoryError, -Djava.security.egd=file:/dev/urandom, -Dlog4j2.isThreadContextMapInheritable=true, -Djruby.regexp.interruptible=true, -Djdk.io.File.enableADOS=true, --add-exports=jdk.compiler/com.sun.tools.javac.api=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.util=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.tree=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.tree.javadoc=ALL-UNNAMED, --add-opens=java.base/java.security=ALL-UNNAMED, --add-opens=java.base/java.io=ALL-UNNAMED, --add-opens=java.base/java.nio.channels=ALL-UNNAMED, --add-opens=java.base/java.nio.channels.spi=ALL-UNNAMED, --add-opens=java.base/java.nio.charset=ALL-UNNAMED, --add-opens=java.base/java.nio.charset.spi=ALL-UNNAMED, --add-opens=java.management/sun.management=ALL-UNNAMED, -Dio.netty.allocator.maxOrder=11]
[2025-04-27T21:46:56.954][INFO ][org.logstash.jackson.StreamReadConstraintsUtil] Jackson default value override `logstash.jackson.stream-read-constraints.max-string-length` configured to `200000000` (logstash default)
[2025-04-27T21:46:56.954][INFO ][org.logstash.jackson.StreamReadConstraintsUtil] Jackson default value override `logstash.jackson.stream-read-constraints.max-number-length` configured to `10000` (logstash default)
[2025-04-27T21:46:56.956][INFO ][org.logstash.jackson.StreamReadConstraintsUtil] Jackson default value override `logstash.jackson.stream-read-constraints.max-nesting-depth` configured to `1000` (logstash default)
[2025-04-27T21:46:56.987][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml' file because command line options are specified
[2025-04-27T21:46:58.483][INFO ][logstash.agent] Successfully started Logstash API endpoint [:port>9600, :ssl_enabled=>false]
[2025-04-27T21:46:58.938][INFO ][org.reflections.Reflections] Reflections took 155 ms to scan 1 urls, producing 149 keys and 521 values
[2025-04-27T21:46:59.397][INFO ][logstash.codecs.jsonlines] ECS compatibility is enabled but 'target' option was not specified. This may cause fields to be set at the top-level of the event where they are likely to clash with the Elastic Common Schema. It is recommended to set the 'target' option to avoid potential schema conflicts (if your data is ECS compliant or non-conflicting, feel free to ignore this message)
[2025-04-27T21:46:59.422][INFO ][logstash.javapipeline] Pipeline `main` is configured with `pipeline.ecs_compatibility: v8` setting. All plugins in this pipeline will default to `ecs_compatibility => v8` unless explicitly configured otherwise.
[2025-04-27T21:46:59.445][INFO ][logstash.filters.csv] [main] ECS compatibility is enabled but 'target' option was not specified. This may cause fields to be set at the top-level of the event where they are likely to clash with the Elastic Common Schema. It is recommended to set the 'target' option to avoid potential schema conflicts (if your data is ECS compliant or non-conflicting, feel free to ignore this message)
[2025-04-27T21:46:59.464][INFO ][logstash.javapipeline] [main] Starting pipeline {:pipeline_id=>"main", "pipeline.workers"=>8, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>50, "pipeline.max_inflight"=>1000, "pipeline.sources"=>["D:/logstash/logstash-9.0.0/logstash.conf"], :thread=>"<Thread:0x415bcd0:D:/logstash/logstash-9.0.0/logs/tash-core/lib/logstash/java_pipeline.rb:138 run"}
[2025-04-27T21:47:00.324][INFO ][logstash.javapipeline] [main] Pipeline Java execution initialization time {"seconds"=>0.86}
[2025-04-27T21:47:00.343][INFO ][logstash.javapipeline] [main] Pipeline started ("pipeline.id"=>"main")
[2025-04-27T21:47:00.348][INFO ][filewatch.observingtail] [main][f4dc329cbe57f3ebcd52b48c646f84a9a3899b1c7b5a6cf0204f6eff9e23abe] START, creating Discoverer, Watch with file and sinceId collections
[2025-04-27T21:47:00.358][INFO ][logstash.agent] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[]}

```

When data is loaded into Snowflake from Logstash pipeline, data is loaded in JSON format. Below screenshot shows raw structure format of data. The raw data is stored in the table called 'CUSTOMERS_RAW'. Snowflake typically stores the JSON data in a Variant Column i.e raw_data. This means the **whole JSON blob** is in 1 field.

```

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_RAW

```

```

    8 CREATE OR REPLACE SCHEMA CUSTOMER_SCHEMA;
    9
    10 USE DATABASE CUSTOMER_DATA;
    11 USE SCHEMA CUSTOMER_SCHEMA;
    12
    13 SELECT *
    14   FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_RAW
    15
  
```

Results

RAW_DATA
{ "@timestamp": "2025-04-28T04:47:00.393899200Z", "@version": "1", "active": true, "customer_id": 6, "customer_name": "Dean King", "email": "new_email@example.com", "join_date": "2025-04-19", "stage": "P", "version": "1" }
{ "@timestamp": "2025-04-28T04:47:00.396891800Z", "@version": "1", "active": true, "customer_id": 14, "customer_name": "Barbara Nelson", "email": "ffraizer@example.org", "join_date": "2021-01-13", "stage": "P", "version": "1" }
{ "@timestamp": "2025-04-28T04:47:00.400881500Z", "@version": "1", "active": true, "customer_id": 22, "customer_name": "Crystal Watson", "email": "Clark@example.com", "join_date": "2023-02-13", "stage": "P", "version": "1" }
{ "@timestamp": "2025-04-28T04:47:00.405477600Z", "@version": "1", "active": true, "customer_id": 30, "customer_name": "Megan Johnson", "email": "scd3.newemail@example.com", "join_date": "2020-06-06", "stage": "P", "version": "1" }
{ "@timestamp": "2025-04-28T04:47:00.412848600Z", "@version": "1", "active": true, "customer_id": 38, "customer_name": "Courtney Smith", "email": "test.scd4_update@example.com", "join_date": "2024-03-04", "stage": "P", "version": "1" }
{ "@timestamp": "2025-04-28T04:47:00.415840900Z", "@version": "1", "active": false, "customer_id": 46, "customer_name": "Caecilia Mitchell", "email": "test.scd4_update@example.com", "join_date": "2022-06-09", "stage": "F", "version": "1" }

Query Details

- Query duration: 848ms
- Rows: 101
- Query ID: 01bc15b3-0105-1f94-0...

RAW_DATA

- @timestamp: 101
- @version: 101
- active: 101

Ask Copilot

Mapping of raw data into structured data

While Snowflake stores the data as JSON by default, a transformation step is required to extract and insert this data into structured tables for querying. This mapping query ensures that JSON data becomes clean, structured, and usable within the database schema. The table 'customers_raw' contains raw JSON data. The below query reads JSON objects and maps/extracts each field into a structured format and inserts into 'customers_parsed' table. The below query is casting datatypes to INT, STRING, BOOLEAN.

```

    16 INSERT INTO customers_parsed
    17   SELECT
    18     CAST(GET_PATH(raw_data, 'customer_id') AS INT) AS customer_id,
    19     CAST(GET_PATH(raw_data, 'customer_name') AS STRING) AS customer_name,
    20     CAST(GET_PATH(raw_data, 'email') AS STRING) AS email,
    21     CAST(
    22       SPLIT_PART(REPLACE(GET_PATH(raw_data, 'join_date')::STRING, "", ""), 'T', 1)
    23     AS DATE
    24   ) AS join_date,
    25   CASE
    26     WHEN LOWER(GET_PATH(raw_data, 'active'))::STRING = 'true' THEN TRUE
    27     ELSE FALSE
    28   END AS active
    29   FROM customers_raw;
    30
    31 SELECT * FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
  
```

Results

CUSTOMER_ID	CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE
6	Dean King	new_email@example.com	2025-04-19	TRUE
14	Barbara Nelson	ffraizer@example.org	2021-01-13	TRUE
22	Crystal Watson	Clark@example.com	2023-02-13	TRUE
30	Megan Johnson	scd3.newemail@example.com	2020-06-06	TRUE
38	Courtney Smith	test.scd4_update@example.com	2024-03-04	TRUE
46	Caecilia Mitchell	test.scd4_update@example.com	2022-06-09	FALSE

Query Details

- Query duration: 164ms
- Rows: 101
- Query ID: 01bc15b3-0105-21df-0...

Ask Copilot

TABLE GENERATION

```
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
```

CUSTOMER_ID	CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE
19	Isaiah Turner	stephanie75@example.org	2020-12-14	FALSE
20	Mackenzie Nguyen	nixonsamatha@example.org	2021-09-07	FALSE
21	Justin Diaz	robinsonandrea@example.com	2021-09-13	TRUE
22	Dean Yang	carriefarmer@example.com	2022-11-25	TRUE
23	Rebecca Higgins	stephaniesmith@example.com	2021-04-21	TRUE
24	Jason Moyer	kmcDonald@example.org	2022-12-10	TRUE
25	Keith Rodriguez	kholtzen@example.org	2022-09-14	TRUE
26	Daniel Dean	mrush@example.net	2024-03-05	FALSE
27	Kendra Frank	robertmorales@example.net	2024-05-07	FALSE
28	Angel Martin	diazaustralia@example.com	2022-09-02	TRUE
29	James Palmer	saraodonell@example.com	2024-06-07	TRUE

DBT INSTALLATION

Dbt helps transform raw data into clean, analysable datasets inside data warehouse. dbt supports incremental loading, testing, and documentation, making data pipelines efficient and reliable. With features like ref(), dbt ensures transformations are easy to manage and scalable.

```
C:\Users\Srilakshmi N Murthy>pip install dbt-core dbt-snowflake
Collecting dbt-core
  Downloading dbt_core-1.8.7-py3-none-any.whl.metadata (3.9 kB)
Collecting dbt-snowflake
  Downloading dbt_snowflake-1.8.4-py3-none-any.whl.metadata (3.2 kB)
Collecting agate<1.10,>=1.7.0 (from dbt-core)
  Downloading agate-1.9.1-py2.py3-none-any.whl.metadata (3.2 kB)
Collecting Jinja2<4,>=3.1.3 (from dbt-core)
  Using cached jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting marshmallow<4.0,>>3.9 (from dbt-core)
  Downloading marshmallow-3.14-py3-none-any.whl.metadata (114 kB)
Collecting logbook<1.6,>=1.5 (from dbt-core)
  Downloading Logbook-1.5.3.tar.gz (85 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting click<9.0,>>8.0.2 (from dbt-core)
  Downloading Click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Collecting networkx<4.0,>=2.3 (from dbt-core)
  Downloading networkx-3.1-py3-none-any.whl.metadata (5.3 kB)
Collecting protobuf<5,>=4.0 (from dbt-core)
  Downloading protobuf-4.25.7-cp38-cp38-win_amd64.whl.metadata (541 bytes)
Collecting requests<3.0.0 (from dbt-core)
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting pathspec<0.13,>=0.9 (from dbt-core)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Collecting sqlparse<0.6.0,>=0.5.0 (from dbt-core)
  Downloading sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
Collecting dbt-extractor<=0.8,>=0.5.0 (from dbt-core)
  Downloading dbt_extractor-0.5.1-cp38-abi3-win_amd64.whl.metadata (4.3 kB)
Collecting minimal-snowplow-tracker<0.1,>=0.0.2 (from dbt-core)
  Downloading minimal-snowplow-tracker-0.0.2.tar.gz (12 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting dbt-semantic-interfaces<0.6,>=0.5.1 (from dbt-core)
  Downloading dbt_semantic_interfaces-0.5.1-py3-none-any.whl.metadata (2.6 kB)
Collecting dbt-common<2.0,>=1.0.4 (from dbt-core)
```

```
Command Prompt - dbt init customer_data_project
20:45:44 [ConfigFolderDirectory]: Unable to parse logging event dictionary. Failed to parse dir field: expected string or bytes-like object.. Dictionary: {'dir': WindowsPa
th('C:/Users/Srilakshmi N Murthy/.dbt')}
20:45:44 Creating dbt configuration folder at
20:45:44 Your new dbt project "customer_data_project" was created!

For more information on how to configure the profiles.yml file,
please consult the dbt documentation here:

https://docs.getdbt.com/docs/configure-your-profile

One more thing:

Need help? Don't hesitate to reach out to us via GitHub issues or on Slack:
https://community.getdbt.com/

Happy modeling!

20:45:44 Setting up your profile.
Which database would you like to use?
[1] snowflake

(Don't see the one you want? https://docs.getdbt.com/docs/available-adapters)

Enter a number: 1
account (https://<this\_value>.snowflakecomputing.com): 
```

The below screenshot explains about the connection between DBT and Snowflake with the snowflake account credentials.

```
Command Prompt
Happy modeling!

20:45:44 Setting up your profile.
Which database would you like to use?
[1] snowflake

(Don't see the one you want? https://docs.getdbt.com/docs/available-adapters)

Enter a number: 1
account (https://<this\_value>.snowflakecomputing.com): ozb70682
user (dev username): SrilakshmiN
[1] password
[2] keypair
[3] sso
Desired authentication type option (enter a number): 1
password (dev password):
role (dev role): ACCOUNTADMIN
warehouse (warehouse name): COMPUTE_WH
database (default database that dbt will build objects in): CUSTOMER_DATA
schema (default schema that dbt will build objects in): CUSTOMER_SCHEMA
threads (1 or more) [1]: 1
21:11:49 Profile customer_data_project written to C:\Users\Srilakshmi N Murthy\.dbt\profiles.yml using target's profile_template.yml and your supplied values. Run 'dbt deb
ug' to validate the connection.

D:\>
```

```
21:15:13 Running with dbt=1.8.7
21:15:13 dbt version: 1.8.7
21:15:13 python version: 3.8.1
21:15:13 python path: c:\python\python38\python.exe
21:15:13 os info: Windows-10-10.0.19041-SP0
21:15:13 Using profiles dir at C:\Users\Srilakshmi N Murthy\.dbt
21:15:13 Using profiles.yml file at C:\Users\Srilakshmi N Murthy\.dbt\profiles.yml
21:15:13 Using dbt_project.yml file at D:\customer_data_project\dbt_project.yml
21:15:13 adapter type: snowflake
21:15:13 adapter version: 1.8.4
21:15:14 Configuration:
21:15:14   profiles.yml file [OK found and valid]
21:15:14   dbt_project.yml file [OK found and valid]
21:15:14 Required dependencies:
21:15:14   - git [OK found]

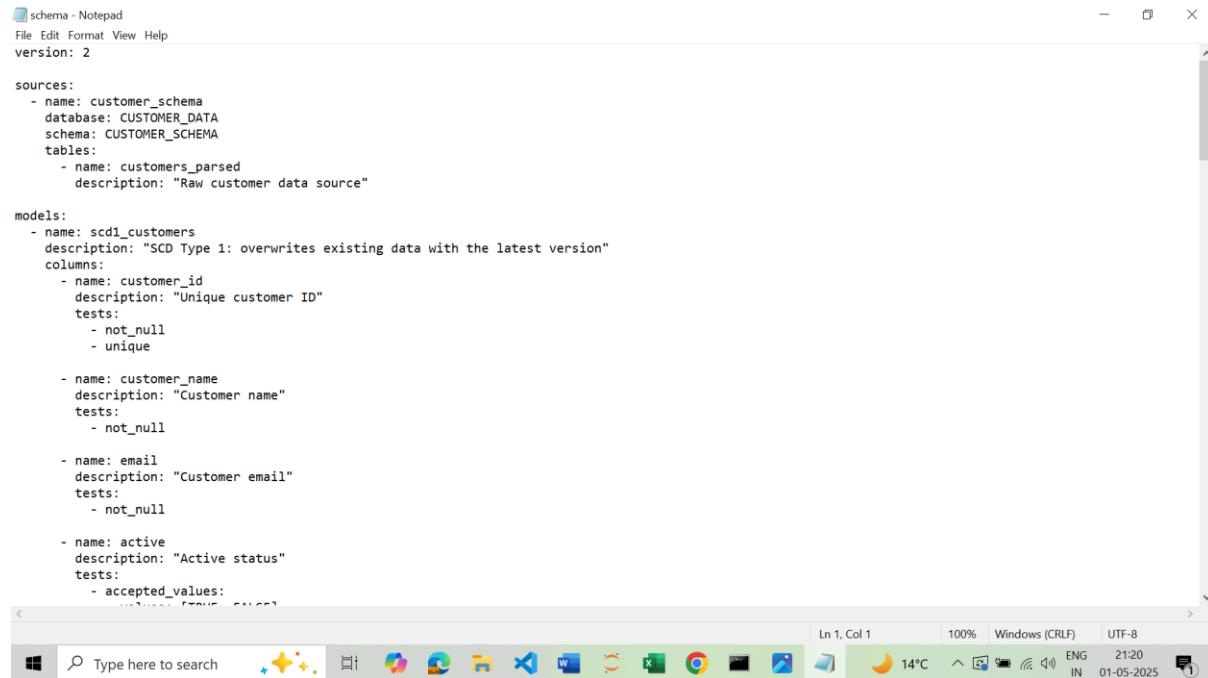
21:15:14 Connection:
21:15:14   account: ozb70682
21:15:14   user: Srilakshmi
21:15:14   database: CUSTOMER_DATA
21:15:14   warehouse: COMPUTE_WH
21:15:14   role: ACCOUNTADMIN
21:15:14   schema: CUSTOMER_SCHEMA
21:15:14   authenticator: None
21:15:14   oauth_client_id: None
21:15:14   query_tag: None
21:15:14   client_session_keep_alive: False
21:15:14   host: None
21:15:14   port: None
21:15:14   proxy_host: None
21:15:14   proxy_port: None
21:15:14   protocol: None
21:15:14   connect_retries: 1
21:15:14   connect_timeout: None
21:15:14   retry_on_database_errors: False
21:15:14   retry_all: False
21:15:14   insecure_mode: False
21:15:14   reuse_connections: None
21:15:14 Registered adapter: snowflake=1.8.4
21:15:15 Connection test: [OK connection ok]
```

```
D:\customer_data_project\models>dbt run
21:47:45  Running with dbt=1.8.7
21:47:46 Registered adapter: snowflake=1.8.4
21:47:47 Found 3 models, 4 data tests, 1 source, 454 macros
21:47:47
21:48:10 Concurrency: 1 threads (target='dev')
21:48:10
21:48:10 1 of 3 START sql view model CUSTOMER_SCHEMA.customer_summary ..... [RUN]
21:48:10 1 of 3 OK created sql view model CUSTOMER_SCHEMA.customer_summary ..... [SUCCESS 1 in 0.84s]
21:48:10 2 of 3 START sql table model CUSTOMER_SCHEMA.my_first_dbt_model ..... [RUN]
21:48:12 2 of 3 OK created sql table model CUSTOMER_SCHEMA.my_first_dbt_model ..... [SUCCESS 1 in 1.63s]
21:48:12 3 of 3 START sql view model CUSTOMER_SCHEMA.my_second_dbt_model ..... [RUN]
21:48:13 3 of 3 OK created sql view model CUSTOMER_SCHEMA.my_second_dbt_model ..... [SUCCESS 1 in 0.85s]
21:48:13
21:48:13 Finished running 2 view models, 1 table model in 0 hours 0 minutes and 26.03 seconds (26.03s).
21:48:13
21:48:13 Completed successfully
21:48:13
21:48:13 Done. PASS=3 WARN=0 ERROR=0 SKIP=0 TOTAL=3

D:\customer_data_project\models>
```

Schema.yml file

The .yml (YAML) file in dbt is used to **document models, columns, and tests** in a structured format. It allows to **add descriptions for each model and column** and define **data quality tests** such as `not_null`, `unique`, and `accepted_values`. The schema.yml file is used to organize tests for all SCD models and provide clear metadata about each table and its columns.



```
schema - Notepad
File Edit Format View Help
version: 2

sources:
  - name: customer_schema
    database: CUSTOMER_DATA
    schema: CUSTOMER_SCHEMA
    tables:
      - name: customers_parsed
        description: "Raw customer data source"

models:
  - name: scd1_customers
    description: "SCD Type 1: overwrites existing data with the latest version"
    columns:
      - name: customer_id
        description: "Unique customer ID"
        tests:
          - not_null
          - unique
      - name: customer_name
        description: "Customer name"
        tests:
          - not_null
      - name: email
        description: "Customer email"
        tests:
          - not_null
      - name: active
        description: "Active status"
        tests:
          - accepted_values:
              - TRUE
              - FALSE
```

MODEL CREATIONS

SCD1 – SCD1 is used to handle changes in dimensional data where the old data is overwritten by the new data. For example, customer email id is updated to ‘new_email@example.com’ for customer id ‘6’ in CUSTOMERS_PARSED table. When we run the dbt for ‘scd1_customers’ details are overwritten without storing any history. **SCD1 (Slowly Changing Dimension Type 1)** overwrites existing dimension data with the latest updates, ensuring that only **the current state is stored**.

```
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED

UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET EMAIL = 'new_email@example.com'
WHERE CUSTOMER_ID = '6';

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD1_CUSTOMERS
```

```

scd1_customers - Notepad
File Edit Format View Help
{{ config(
    materialized='incremental',
    unique_key='customer_id'
) }}

{% if is_incremental() %}

with source as (
    select * from {{ source('customer_schema', 'customers_parsed') }}
),
existing as (
    select * from {{ this }}
),
final as (
    select
        s.customer_id,
        s.customer_name,
        s.email,
        s.join_date,
        s.active
    from source s
    left join existing e on s.customer_id = e.customer_id
    where e.customer_id is null
        or md5(s.customer_name || s.email || s.join_date || s.active)
            != md5(e.customer_name || e.email || e.join_date || e.active)
)
select * from final

{% else %}

select
    customer_id,
    customer_name,
    ...
)

```

Notepad window showing the 'scd1_customers' script. The status bar at the bottom right shows: Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8 | 14°C | ENG IN | 01-05-2025.

Customers_parsed table is updated.

Snowflake UI screenshot showing the CUSTOMERS_PARSED table in the CUSTOMER_SCHEMA schema. The table has 101 rows. A query is being run to update the EMAIL column for a specific customer ID. The results show 1 row updated. The status bar at the bottom right shows: 26°C | ENG IN | 30-04-2025.

#	CUSTOMER_ID	NUMBER(38,0)
A	CUSTOMER_NAME	VARCHAR(16777216)
A	EMAIL	VARCHAR(16777216)
(1)	JOIN_DATE	DATE
0 1	ACTIVE	BOOLEAN

Screenshot of the Snowflake Query Editor interface showing a query running on the CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_RAW table.

Query:

```

197 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_RAW
198
199
200 SELECT *
201 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD1_CUSTOMERS
202
203 SELECT *
204 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
205
206 UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
207 SET EMAIL = 'new_email@example.com'
208 WHERE CUSTOMER_ID = '6';
209
210
211

```

Results:

CUSTOMER_ID	CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE
1	Dean King	new_email@example.com	2025-04-19	TRUE
2	Barbara Nelson	ffrazier@example.org	2021-01-13	TRUE
3	Crystal Watson	keith23@example.org	2023-02-13	TRUE
4	Megan Johnson	nfernandez@example.net	2020-06-06	TRUE
5	Courtney Smith	charlesneil@example.org	2024-03-04	TRUE
6	Cassidy Mitchell	fmartin@example.com	2022-06-09	FALSE

Query Details:

- Query duration: 64ms
- Rows: 101
- Query ID: 01bc0b3e-0105-1ade-0...

Dbt is running for SCD1 Model

Screenshot of a Command Prompt window showing the execution of a dbt run command for the SCD1 model.

```

D:\customer_data_project>dbt run --select scd1_customers
22:22:30  Running with dbt=1.8.7
22:22:31  Registered adapter: snowflake=1.8.4
22:22:31  Found 7 models, 4 data tests, 1 source, 454 macros
22:22:31
22:22:33  Concurrency: 1 threads (target='dev')
22:22:33
22:22:33  1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd1_customers ..... [RUN]
22:22:37  1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd1_customers ..... [SUCCESS 1 in 3.92s]
22:22:37
22:22:37  Finished running 1 incremental model in 0 hours 0 minutes and 5.93 seconds (5.93s).
22:22:37
22:22:37  Completed successfully
22:22:37
22:22:37  Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>

```

The screenshot shows the Snowflake web interface. On the left, the sidebar displays the database structure under 'MY_SECOND_DB_MODEL'. In the center, a query is being run:

```

197 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_RAW
198
199
200 SELECT *
201 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD1_CUSTOMERS
202
203 SELECT *
204 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
205
206 UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
207 SET EMAIL = 'new_email@example.com'
208 WHERE CUSTOMER_ID = '6';
209
210
211

```

The results pane shows a table with 101 rows, listing customer details. The table has columns: #, CUSTOMER_ID, CUSTOMER_NAME, EMAIL, JOIN_DATE, and ACTIVE.

#	CUSTOMER_ID	CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE
1	6	Dean King	new_email@example.com	2025-04-19	TRUE
2	14	Barbara Nelson	ffrazier@example.org	2021-01-13	TRUE
3	22	Crystal Watson	keith23@example.org	2023-02-13	TRUE
4	30	Megan Johnson	nhernandez@example.net	2020-06-06	TRUE
5	38	Courtney Smith	charlesneil@example.org	2024-03-04	TRUE
6	46	Cassidy Mitchell	fmartin@example.com	2022-06-09	FALSE

On the right, the 'Query Details' panel shows the duration as 152ms and the number of rows as 101.

SCD2 - SCD2 (Slowly Changing Dimension Type 2) enables **full history tracking by inserting new rows** when changes occur, while keeping existing records unchanged. For example, customers email id is updated to '`keith.final_test_123@example.com`' from '`keith23@example.com`' for `customer_id = '22'`. So SCD2 ensures to keep track of history by inserting new row by updating only that field. It keeps all others as it is. Hence it changes the flag to '**FALSE**' for the old record and changes to '**TRUE**' for new updated record.

```

UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'keith.final_test_123@example.com'
WHERE customer_id = 22;

```

SELECT * FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD2_CUSTOMERS;

```

sod2_customers - Notepad
File Edit Format View Help
{{ config(
    materialized='incremental',
    unique_key='customer_id', load_date'
) }}

with source as (
    select * from {{ source('customer_schema', 'customers_parsed') }}
)

{% if is_incremental() %}

, current_customers as (
    select
        c.customer_id,
        c.customer_name,
        c.email,
        c.join_date,
        c.active,
        c.load_date,
        c.current_flag
    from {{ this }} as c
    where c.current_flag = true
)

, new_customers as (
    select
        s.customer_id,
        s.customer_name,
        s.email,
        s.join_date,
        s.active,
        current_timestamp() as load_date,
        true as current_flag
    from source as s
)

```

The code is a complex SQL script for an incremental load into a SCD2 dimension. It uses the Snowflake configuration block to define the materialized view and unique key. It then defines two temporary tables: 'current_customers' and 'new_customers'. The 'current_customers' table selects columns from the source table where the 'current_flag' is true. The 'new_customers' table selects columns from the source table, setting the 'load_date' to the current timestamp and 'current_flag' to true. This logic ensures that only changed rows are inserted into the dimension table.

```
03:26:26     Database Error in model scd2_customers (models\scd2_customers.sql)
002028 (42601): SQL compilation error:
ambiguous column name 'LOAD_DATE'
compiled code at target\run\customer_data_project\models\scd2_customers.sql
03:26:26
03:26:26 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1

D:\customer_data_project>dbt run --select scd2_customers --full-refresh
03:29:12     Running with dbt=1.8.7
03:29:13 Registered adapter: snowflake=1.8.4
03:29:14 Found 8 models, 4 data tests, 1 source, 454 macros
03:29:14
03:29:15     Concurrency: 1 threads (target='dev')
03:29:15
03:29:15 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd2_customers ..... [RUN]
03:29:17 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd2_customers ..... [SUCCESS 1 in 1.91s]
03:29:17
03:29:17 Finished running 1 incremental model in 0 hours 0 minutes and 3.43 seconds (3.43s).
03:29:17
03:29:17 Completed successfully
03:29:17
```

The screenshot shows the Snowflake web interface. The top navigation bar includes tabs for 'snowflake - Google Search', 'HW#9', 'Snowflake Account Login Guide', and '2025-04-27 10:26pm - Snowflake'. The main area displays a query editor with the following details:

- Databases:** CUSTOMER_DATA.CUSTOMER_SCHEMA
- Worksheets:** Settings
- Pinned Objects:** MY_SECOND_DBT_MODEL
- Object Tree:** MY_SECOND_DBT_MODEL contains SCD1_CUSTOMERS, SCD2_CUSTOMERS, Stages (including CUSTOMER_STAGE), INFORMATION_SCHEMA, and PUBLIC.
- Table Preview:** CUSTOMERS_PARSED (101 Rows) with columns: #, CUSTOMER_ID (NUMBER(38,0)), CUSTOMER_NAME (VARCHAR(16777216)), EMAIL (VARCHAR(16777216)), JOIN_DATE (DATE), and ACTIVE (BOOLEAN).
- Query Editor:** An UPDATE statement is being run:

```
UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'keith.final_test.123@example.com'
WHERE customer_id = 22;
```
- Results:** Shows 1 row updated and 1 multi-joined row updated.
- Query Details:** Duration: 388ms, Rows: 1, Query ID: 01bc0c82-0105-1d07-0...
- Bottom Bar:** Includes a search bar, file icons, and system status indicators.

The screenshot shows the Snowflake web interface. The top navigation bar displays the URL `app.snowflake.com/vrhiae/bgb80107/w2WMhN340Mcx#query`. The left sidebar contains pinned databases: `MY_SECOND_DB_MODEL`, `INFORMATION_SCHEMA`, and `PUBLIC`. The main area shows a query editor with the following code:

```
L24
225
226
227
228
229
230 SELECT * FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD2_CUSTOMERS
231 WHERE customer_id = '22'
232
233
234
235
236 UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
237 SET email = 'keith.final_test_123@example.com'
238
239
```

The results table below shows two rows of data:

CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE	LOAD_DATE	CURRENT_FLAG
rstal Watson	keith.final_test_123@example.com	2023-02-13	TRUE	2025-04-30 20:34:31.962 -0700	TRUE
rstal Watson	keith23@example.com	2023-02-13	TRUE	2025-04-30 19:47:00.000 -0700	FALSE

On the right side, the "Query Details" panel shows a duration of 92ms and 2 rows processed. The bottom right corner features the "Ask Copilot" button.

```

ambiguous column name 'LOAD_DATE'
  compiled code at target\run\customer_data_project\models\scd2_customers.sql
03:29:43 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1

D:\customer_data_project>dbt run --select scd2_customers --full-refresh
03:34:28 Running with dbt=1.8.7
03:34:29 Registered adapter: snowflake=1.8.4
03:34:29 Found 8 models, 4 data tests, 1 source, 454 macros
03:34:29
03:34:31 Concurrency: 1 threads (targets='dev')
03:34:31
03:34:31 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd2_customers ..... [RUN]
03:34:33 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd2_customers ..... [SUCCESS 1 in 1.96s]
03:34:33
03:34:33 Finished running 1 incremental model in 0 hours 0 minutes and 3.35 seconds (3.35s).
03:34:33
03:34:33 Completed successfully
03:34:33
03:34:33 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1

```

17°C ENG IN 20:48 30-04-2025

SCD3 - SCD3 is about tracking limited history by adding new columns rather than adding new rows. Instead of keeping **multiple rows per record** (like SCD2), SCD3 **stores the current value and one (or a few) previous values in additional columns**. For example, email is changed to '`scd3.newemail@example.com`' from '`n hernandez@example.com`'. It created a new column named '`prev_email`' and stored the history instead of creating new row. So **the row stays the same (no new row is added)**, but the **columns are updated to reflect the change**.

UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED

```

SET email = 'scd3.newemail@example.com'
WHERE customer_id = 30;

```

SELECT * FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD3_CUSTOMERS

```

WHERE customer_id = 30;

```

```

scd3_customers - Notepad
File Edit Format View Help
{{ config(
    materialized='incremental',
    unique_key='customer_id'
) }}

with source as (
    select * from {{ source('customer_schema', 'customers_parsed') }}
)

{% if is_incremental() %}

, current_records as (
    select * from {{ this }}
)

, merged as (
    select
        s.customer_id,
        s.customer_name,
        s.email,
        s.join_date,
        s.active,
        case
            when s.email != c.email then c.email
            else c.prev_email
        end as prev_email
    from source s
    left join current_records c
        on s.customer_id = c.customer_id
)

select * from merged
{% else %}

```

Ln 1, Col 1 100% Windows (CRLF) UTF-8
14°C ENG IN 21:22 01-05-2025

```

C:\ Command Prompt
03:59:38 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1
D:\customer_data_project>dbt run --select scd3_customers --full-refresh
03:59:38 Running with dbt=1.8.7
03:59:35 Registered adapter: snowflake=1.8.4
03:59:36 Found 8 models, 4 data tests, 1 source, 454 macros
03:59:36
03:59:38 Concurrency: 1 threads (target='dev')
03:59:38
03:59:38 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd3_customers ..... [RUN]
03:59:40 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd3_customers ..... [SUCCESS 1 in 2.06s]
03:59:40
03:59:40 Finished running 1 incremental model in 0 hours 0 minutes and 4.25 seconds (4.25s).
03:59:40
03:59:40 Completed successfully
03:59:40
03:59:40 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>dbt run --select scd3_customers
04:00:37 Running with dbt=1.8.7
04:00:38 Registered adapter: snowflake=1.8.4
04:00:39 Found 8 models, 4 data tests, 1 source, 454 macros
04:00:39
04:00:40 Concurrency: 1 threads (target='dev')
04:00:40
04:00:40 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd3_customers ..... [RUN]
04:00:43 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd3_customers ..... [SUCCESS 101 in 3.28s]
04:00:43
04:00:43 Finished running 1 incremental model in 0 hours 0 minutes and 4.60 seconds (4.60s).
04:00:43
04:00:43 Completed successfully
04:00:43
04:00:43 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>

```

The screenshot shows the Snowflake web interface. On the left, the sidebar displays the database structure under 'MY_SECOND_DB_MODEL'. It includes a tree view of tables like 'SCD1_CUSTOMERS' and 'SCD2_CUSTOMERS', and a 'Stages' section containing 'CUSTOMER_STAGE'. Below this is a table named 'CUSTOMERS_PARSED' with 101 rows. On the right, a query editor window is open, showing a SQL UPDATE statement:

```

UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'scd3.newemail@example.com'
WHERE customer_id = 30;

```

The results tab shows the execution details:

# number of rows updated	# number of multi-joined rows updated
1	1

Query Details:

- Query duration: 616ms
- Rows: 1
- Query ID: 01bc0c90-0105-1ac8-0...

```

2025-04-27 4:13pm 2025-04-27 10:26pm + ⌂
CUSTOMER_DATA.CUSTOMER_SCHEMA Settings
307
308
309
310
311 UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
312 SET email = 'scd3.newemail@example.com'
313 WHERE customer_id = 30;
314
315
316
317
318 SELECT * FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD3_CUSTOMERS
319 WHERE customer_id = 30;
320
321
RESULTS
# CUSTOMER_ID CUSTOMER_NAME EMAIL JOIN_DATE ACTIVE PREV_EMAIL
1 30 Megan Johnson scd3.newemail@example.com 2020-06-06 TRUE nherandez@example.com

```

Query Details
Query duration 130ms
Rows 1
Query ID 01bc0e91-0105-1d00-0...
Show more Ask Copilot
CUSTOMER_ID

SCD4 - SCD4 is called a Hybrid approach because it **combines SCD1 (overwrite) with a separate history table**. Current data in one table is maintained (like SCD1 — overwrite the existing row). But changes are **archived into a separate history table** every time there's a change.

For example, customers email is updated to '`test.scd4_update@example.com`' in `CUSTOMERS_PARSED` from '`Clark@example.com`'. It creates separate history table with the old data. While in current table, it retains the latest changes for the same record. The `scd4_customers_current` table stores only the latest version of each customer's record, similar to SCD1. The `scd4_customers_history` table captures every previous version of the customer's data before updates.

```

UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'test.scd4_update@example.com'
WHERE customer_id = 70;

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_CURRENT

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_HISTORY

```

```

C:\ Command Prompt
18:48:16 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1
D:\customer_data_project>dbt run --select scd4_customers_current
18:51:42 Running with dbt=1.8.7
18:51:43 Registered adapter: snowflake=1.8.4
18:51:44 Found 8 models, 22 data tests, 1 source, 454 macros
18:51:44 Concurrency: 1 threads (target='dev')
18:51:45 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd4_customers_current ..... [RUN]
18:51:46 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd4_customers_current . [SUCCESS 1 in 1.66s]
18:51:46 Finished running 1 incremental model in 0 hours 0 minutes and 2.83 seconds (2.83s).
18:51:47 Completed successfully
18:51:47 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>dbt run --select scd4_customers_history
18:52:40 Running with dbt=1.8.7
18:52:41 Registered adapter: snowflake=1.8.4
18:52:42 Found 8 models, 22 data tests, 1 source, 454 macros
18:52:42 Concurrency: 1 threads (target='dev')
18:52:43 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd4_customers_history ..... [RUN]
18:52:44 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd4_customers_history . [SUCCESS 1 in 1.53s]
18:52:44 Finished running 1 incremental model in 0 hours 0 minutes and 2.60 seconds (2.60s).
18:52:44 Completed successfully
18:52:44 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>

```

The screenshot shows the Snowflake web interface. On the left, there's a sidebar with navigation icons and a search bar. The main area has two tabs: 'Results' and 'Chart'. The 'Results' tab is active, showing a table with one row. The table has two columns: '# number of rows updated' and '# number of multi-joined rows updated'. The first column has a value of 1, and the second column has a value of 0. To the right of the table, there's a 'Query Details' section with a progress bar for 'Query duration' (530ms), a row count of 1, and a query ID of 01bc100f-0105-1f60-00... Below this, there's an 'Ask Copilot' button.

# number of rows updated	# number of multi-joined rows updated
1	0

Query Details

- Query duration: 530ms
- Rows: 1
- Query ID: 01bc100f-0105-1f60-00...

Ask Copilot

app.snowflake.com/vrhiaen/bgb80107/w2WMhN340Mcx#query

Databases Worksheets

2025-04-27 4:13pm 2025-04-27 10:26pm + ⏪

ACCOUNTADMIN COMPUTE_WH (X-Small) Share ⏪

Code Versions ⏪

Pinned (1)

MY_SECOND_DBT_MODEL

Search objects

CUSTOMER_DATA

CUSTOMER_SCHEMA

Tables

- CUSTOMERS_PARSED
- CUSTOMERS_RAW
- SCD1_CUSTOMERS
- SCD2_CUSTOMERS
- SCD3_CUSTOMERS
- SCD4_CUSTOMERS_CURRENT
- SCD4_CUSTOMERS_HISTORY

Views

Stages

INFORMATION_SCHEMA

PUBLIC

SNOWFLAKE

SNOWFLAKE_LEARNING_DB

CUSTOMER_DATA.CUSTOMER_SCHEMA Settings ⏪

369
370
371
372
373
374
375
376
377
378
379
380
381
382
383

```
SELECT *  
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_CURRENT
```

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_HISTORY

Results ⏪ Chart

#	CUSTOMER_ID	CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE	LOAD_DATE
8	62	Mr. Jamie McKee	sharpmichelle@example.org	2025-04-07	FALSE	2025-05-01 11
9	70	Ellen Clark	test.scd4_update@example.c	2020-10-04	TRUE	2025-05-01 11
10	78	Bradley Ruiz	johnhuerta@example.net	2022-02-15	FALSE	2025-05-01 11
11	86	Lauren Weeks	danielfisher@example.com	2021-03-17	FALSE	2025-05-01 11
12	94	Jonathan Juarez	susannolen@example.com	2023-01-10	TRUE	2025-05-01 11

Query Details

Query duration 356ms

Rows 101

Query ID 01bc1010-0105-1f8c-00...

Show more ⏪ Ask Copilot

Type here to search

20 20°C ENG IN 11:57 01-05-2025

app.snowflake.com/vrhiaen/bgb80107/w2WMhN340Mcx#query

Databases Worksheets

2025-04-27 4:13pm 2025-04-27 10:26pm + ⏪

ACCOUNTADMIN COMPUTE_WH (X-Small) Share ⏪

Code Versions ⏪

Pinned (1)

MY_SECOND_DBT_MODEL

Search objects

CUSTOMER_DATA

CUSTOMER_SCHEMA

Tables

- CUSTOMERS_PARSED
- CUSTOMERS_RAW
- SCD1_CUSTOMERS
- SCD2_CUSTOMERS
- SCD3_CUSTOMERS
- SCD4_CUSTOMERS_CURRENT
- SCD4_CUSTOMERS_HISTORY

Views

Stages

INFORMATION_SCHEMA

PUBLIC

SNOWFLAKE

SNOWFLAKE_LEARNING_DB

CUSTOMER_DATA.CUSTOMER_SCHEMA Settings ⏪

372
373
374
375
376
377
378
379
380
381
382
383
384
385
386

```
SELECT *  
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_CURRENT
```

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_HISTORY

Results ⏪ Chart

#	CUSTOMER_ID	CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE	ARCH	Query Details
1	70	Ellen Clark	Clark@example.com	2020-10-04	true	2025-05	Query duration 76ms

Rows 1

Query ID 01bc101e-0105-1ac9-00...

Show more ⏪ Ask Copilot

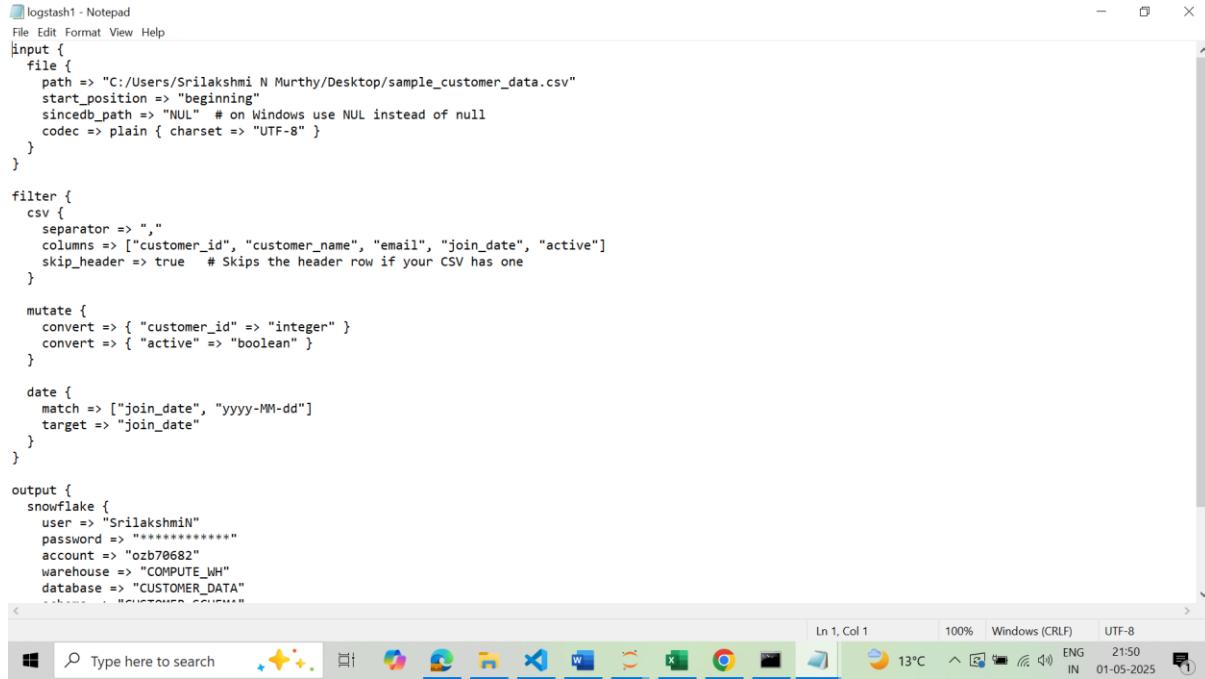
CUSTOMER_ID A

Type here to search

22 22°C ENG IN 12:13 01-05-2025

Error handling and Logging

1. In this project, proper error handling and logging are crucial for ensuring data quality and smooth troubleshooting. DBT automatically logs **detailed compilation, run, and test logs**. If any error occurs (e.g., SQL compilation errors, missing tables, type mismatches), DBT **captures and reports**.
2. Validation tests serve as automatic error detections. Any **unexpected nulls, duplicates, or invalid values** trigger clear test failures.
3. Incremental model safety has been taken to avoid errors if any new changes are done in the records.
4. Error Prevention practices are used in code files, for example `skip_header=>true` to avoid misinterpreting CSV headers as data.
5. Logstash logging : Used `since_db_path => "NUL"` and `start_position => "beginning"` to ensure no **duplicate processing errors** happen for repeated runs.



```
logstash1 - Notepad
File Edit Format View Help
input {
  file {
    path => "C:/Users/Srilakshmi N Murthy/Desktop/sample_customer_data.csv"
    start_position => "beginning"
    since_db_path => "NUL" # on Windows use NUL instead of null
    codec => plain { charset => "UTF-8" }
  }
}

filter {
  csv {
    separator => ","
    columns => ["customer_id", "customer_name", "email", "join_date", "active"]
    skip_header => true # Skips the header row if your CSV has one
  }
}

mutate {
  convert => { "customer_id" => "integer" }
  convert => { "active" => "boolean" }
}

date {
  match => ["join_date", "yyyy-MM-dd"]
  target => "join_date"
}

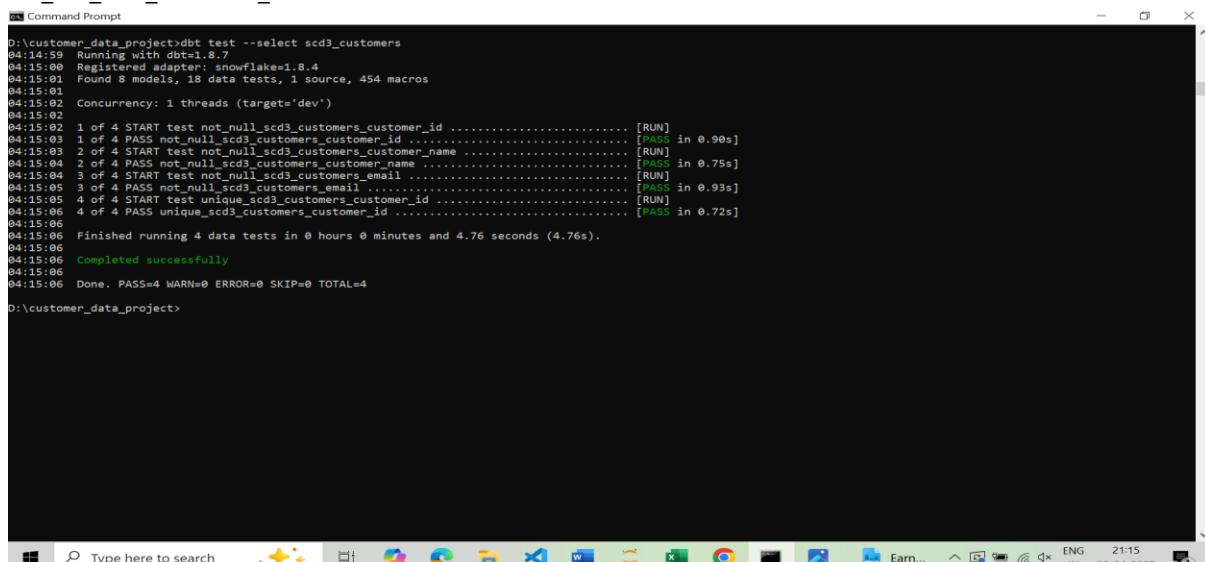
output {
  snowflake {
    user => "SrilakshmiN"
    password => "*****"
    account => "ozb70682"
    warehouse => "COMPUTE_WH"
    database => "CUSTOMER_DATA"
    schema => "CUSTOMERS_SCHEMA"
  }
}

```

Below scripts shows all the test cases for scd1, scd2, scd3, scd4. It represents the data tests for particular fields in those particular SCDS.

DBTs built in test cases

`not_null_scd3_customers_customer_id` : Ensures every customer has customer id
`not_null_scd3_customers_customer_name`: Confirms all customers have names
`not_null_scd3_customers_email`: verifies all customers have email and etc



```
D:\customer_data_project>dbt test --select scd3_customers
04:14:59 Running with dbt=1.8.7
04:15:00 Registered adapter: snowflake=1.8.4
04:15:01 Found 8 models, 18 data tests, 1 source, 454 macros
04:15:01
04:15:02 Concurrency: 1 threads (target='dev')
04:15:02
04:15:02 1 of 4 START test not_null_scd3_customers_customer_id ..... [RUN]
04:15:03 1 of 4 PASS not_null_scd3_customers_customer_id ..... [PASS in 0.90s]
04:15:04 2 of 4 START test not_null_scd3_customers_customer_name ..... [RUN]
04:15:05 2 of 4 PASS not_null_scd3_customers_customer_name ..... [PASS in 0.75s]
04:15:04 3 of 4 START test not_null_scd3_customers_email ..... [RUN]
04:15:05 3 of 4 PASS not_null_scd3_customers_email ..... [PASS in 0.93s]
04:15:05 4 of 4 START test unique_scd3_customers_customer_id ..... [RUN]
04:15:06 4 of 4 PASS unique_scd3_customers_customer_id ..... [PASS in 0.72s]
04:15:06
04:15:06 Finished running 4 data tests in 0 hours 0 minutes and 4.76 seconds (4.76s).
04:15:06
04:15:06 Completed successfully
04:15:06
04:15:06 Done. PASS=4 WARN=0 ERROR=0 SKIP=0 TOTAL=4
04:15:06
D:\customer_data_project>
```

```
Command Prompt
04:27:21
04:27:21 Done. PASS=5 WARN=0 ERROR=1 SKIP=0 TOTAL=6

D:\customer_data_project>dbt test --select scd2_customers
04:29:03 Running with dbt=1.8.7
04:29:04 Registered adapter: snowflake=1.8.4
04:29:05 [WARNING]: Deprecated functionality
The 'tests' config has been renamed to 'data_tests'. Please see
https://docs.getdbt.com/docs/build/data-tests#new-data_tests-syntax for more
information.
04:29:05 Found 8 models, 17 data tests, 1 source, 454 macros
04:29:05
04:29:06 Concurrency: 1 threads (target='dev')
04:29:06
04:29:06 1 of 5 START test accepted_values_scd2_customers_current_flag_True_False ..... [RUN]
04:29:06 1 of 5 PASS accepted_values_scd2_customers_current_flag_True_False ..... [PASS in 0.71s]
04:29:06 2 of 5 START test not_null_scd2_customers_customer_id ..... [RUN]
04:29:07 2 of 5 PASS not_null_scd2_customers_customer_id ..... [PASS in 0.77s]
04:29:07 3 of 5 START test not_null_scd2_customers_customer_name ..... [RUN]
04:29:08 3 of 5 PASS not_null_scd2_customers_customer_name ..... [PASS in 0.55s]
04:29:08 4 of 5 START test not_null_scd2_customers_email ..... [RUN]
04:29:08 4 of 5 PASS not_null_scd2_customers_email ..... [PASS in 0.60s]
04:29:08 5 of 5 START test not_null_scd2_customers_load_date ..... [RUN]
04:29:09 5 of 5 PASS not_null_scd2_customers_load_date ..... [PASS in 0.62s]
04:29:09
04:29:09 Finished running 5 data tests in 0 hours 0 minutes and 3.95 seconds (3.95s).
04:29:09
04:29:09 Completed successfully
04:29:09
04:29:09 Done. PASS=5 WARN=0 ERROR=0 SKIP=0 TOTAL=5

D:\customer_data_project>
```

```
Command Prompt
04:14:39 Running with dbt=1.8.7
04:14:40 Registered adapter: snowflake=1.8.4
04:14:41 [WARNING]: Deprecated functionality
The 'tests' config has been renamed to 'data_tests'. Please see
https://docs.getdbt.com/docs/build/data-tests#new-data_tests-syntax for more
information.
04:14:41 Found 8 models, 18 data tests, 1 source, 454 macros
04:14:41
04:14:42 Concurrency: 1 threads (target='dev')
04:14:42
04:14:42 1 of 4 START test not_null_scd1_customers_customer_id ..... [RUN]
04:14:44 1 of 4 PASS not_null_scd1_customers_customer_id ..... [PASS in 1.55s]
04:14:44 2 of 4 START test not_null_scd1_customers_customer_name ..... [RUN]
04:14:44 2 of 4 PASS not_null_scd1_customers_customer_name ..... [PASS in 0.86s]
04:14:44 3 of 4 START test not_null_scd1_customers_email ..... [RUN]
04:14:45 3 of 4 PASS not_null_scd1_customers_email ..... [PASS in 0.87s]
04:14:45 4 of 4 START test unique_scd1_customers_customer_id ..... [RUN]
04:14:46 4 of 4 PASS unique_scd1_customers_customer_id ..... [PASS in 1.14s]
04:14:46
04:14:46 Finished running 4 data tests in 0 hours 0 minutes and 5.37 seconds (5.37s).
04:14:47
04:14:47 Completed successfully
04:14:47
04:14:47 Done. PASS=4 WARN=0 ERROR=0 SKIP=0 TOTAL=4

D:\customer_data_project>dbt test --select scd2_customers
04:14:47 Type here to search  +---+  ENG 21:29 IN 30-04-2025 1
04:14:47
```

```
Command Prompt
18:56:43
18:56:43 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1

D:\customer_data_project>dbt test --select scd4_customers_current
20:15:25 Running with dbt=1.8.7
20:15:26 Registered adapter: snowflake=1.8.4
20:15:27 [WARNING]: Deprecated functionality
The 'tests' config has been renamed to 'data_tests'. Please see
https://docs.getdbt.com/docs/build/data-tests#new-data_tests-syntax for more
information.
20:15:28 Found 8 models, 32 data tests, 1 source, 454 macros
20:15:28
20:15:29 Concurrency: 1 threads (target='dev')
20:15:29
20:15:29 1 of 5 START test accepted_values_scd4_customers_current_active_True_False ... [RUN]
20:15:31 1 of 5 PASS accepted_values_scd4_customers_current_active_True_False ..... [PASS in 1.48s]
20:15:31 2 of 5 START test not_null_scd4_customers_current_customer_id ..... [RUN]
20:15:31 2 of 5 PASS not_null_scd4_customers_current_customer_id ..... [PASS in 0.76s]
20:15:31 3 of 5 START test not_null_scd4_customers_current_customer_name ..... [RUN]
20:15:32 3 of 5 PASS not_null_scd4_customers_current_customer_name ..... [PASS in 1.14s]
20:15:32 4 of 5 START test not_null_scd4_customers_current_email ..... [RUN]
20:15:33 4 of 5 PASS not_null_scd4_customers_current_email ..... [PASS in 0.78s]
20:15:33 5 of 5 START test unique_scd4_customers_current_customer_id ..... [RUN]
20:15:34 5 of 5 PASS unique_scd4_customers_current_customer_id ..... [PASS in 0.65s]
20:15:34
20:15:34 Finished running 5 data tests in 0 hours 0 minutes and 5.99 seconds (5.99s).
20:15:34
20:15:34 Completed successfully
20:15:34
20:15:34 Done. PASS=5 WARN=0 ERROR=0 SKIP=0 TOTAL=5

D:\customer_data_project>
```

VIEW – A view named ‘latest_customers’ is created to keep track of latest records of only Active customers. When the below query is ran, we do not get any records as it keeps only ‘ACTIVE’ records in the view.

```
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.LATEST_CUSTOMERS
WHERE CURRENT_FLAG = 'FALSE'
```

```
D:\customer_data_project>dbt run --select latest_customers
05:49:32+ Running with dbt=1.8.7
05:49:33+ Registered adapter: snowflake=1.8.4
05:49:39+ Found 8 models, 17 data tests, 1 source, 454 macros
05:49:32 Concurrency: 1 threads (target='dev')
05:49:32
05:49:32 1 of 1 START sql view model CUSTOMER_SCHEMA.latest_customers ..... [RUN]
05:49:33 1 of 1 OK created sql view model CUSTOMER_SCHEMA.latest_customers ..... [SUCCESS 1 in 1.48s]
05:49:33
05:49:33 Finished running 1 view model in 0 hours 0 minutes and 4.03 seconds (4.03s).
05:49:33
05:49:33 Completed successfully
05:49:33
05:49:33 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>
```

	CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE	LOAD_DATE	CURRENT_FLAG
1	Dean King	new_email@example.com	2025-04-19	TRUE	2025-04-30 20:34:31.962 -0700	TRUE
2	Barbara Nelson	ffrazier@example.org	2021-01-13	TRUE	2025-04-30 20:34:31.962 -0700	TRUE
3	Crystal Watson	keith.final_test_123@example.org	2023-02-13	TRUE	2025-04-30 20:34:31.962 -0700	TRUE
4	Megan Johnson	rfernandez@example.net	2020-06-06	TRUE	2025-04-30 20:34:31.962 -0700	TRUE
5	Courtney Smith	charlesneil@example.org	2024-03-04	TRUE	2025-04-30 20:34:31.962 -0700	TRUE
6	Cassidy Mitchell	fmartin@example.com	2022-06-09	FALSE	2025-04-30 20:34:31.962 -0700	TRUE

The screenshot shows the Snowflake web interface. The left sidebar lists databases and worksheets. A query titled 'Pinned (1)' is running, showing the following SQL code:

```

    FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.LATEST_CUSTOMERS
    WHERE CURRENT_FLAG = 'FALSE'

    SELECT *
    FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.LATEST_CUSTOMERS
    WHERE CURRENT_FLAG = 'FALSE'

```

The results pane shows a table with columns: CUSTOMER_ID, CUSTOMER_NAME, EMAIL, JOIN_DATE, ACTIVE, LOAD_DATE, and CURRENT_FLAG. The message 'Query produced no results' is displayed.

On the right, the 'Query Details' panel shows: Query duration 167ms, Rows 0, and Query ID 01bc0d09-0105-4d97-0...

Optimization Techniques Used

1. Incremental models: Incremental models are developed for the SCDs (is_incremental() blocks). This means **only new or changed records are processed** during subsequent runs. **This reduces execution time and warehouse cost significantly when compared to full table rebuilds.**

```

scd2_customers - Notepad
File Edit Format View Help
{{ config(
    materialized='incremental',
    unique_key='customer_id, load_date'
) }}

with source as (
    select * from {{ source('customer_schema', 'customers_parsed') }}
)

{% if is_incremental() %}

, current_customers as (
    select
        c.customer_id,
        c.customer_name,
        c.email,
        c.join_date,
        c.active,
        c.load_date,
        c.current_flag
    from {{ this }} as c
    where c.current_flag = true
)

```

2. Source filtering – In SCD SQL logic, source data is filtered to include only new records or history records based on SCD logic. For example, SCD4_history stores history of only changed rows rather than storing whole table

The screenshot shows the Snowflake web interface. The left sidebar lists databases and worksheets. A query titled 'Pinned (1)' is running, showing the following SQL code:

```

    SELECT *
    FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_CURRENT

    SELECT *
    FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_HISTORY

```

The results pane shows a table with columns: CUSTOMER_ID, CUSTOMER_NAME, EMAIL, JOIN_DATE, and ACTIVE. One row is shown:

CUSTOMER_ID	CUSTOMER_NAME	EMAIL	JOIN_DATE	ACTIVE
70	Ellen Clerk	Clark@example.com	2020-10-04	true

On the right, the 'Query Details' panel shows: Query duration 76ms, Rows 1, and Query ID 01bc101e-0105-4ac8-00...

3. Efficient joins and Indexing – To keep the queries simple, only simple and fewer joins are used in SCDs to improve performance. There are no extra cross joins. They are very limited to reduce costs on joins.

```

        . . . . .,
        c.join_date,
        c.active,
        c.load_date,
        c.current_flag
    from { this } as c
    where c.current_flag = true
)

, new_customers as (
    select
        s.customer_id,
        s.customer_name,
        s.email,
        s.join_date,
        s.active,
        current_timestamp() as load_date,
        true as current_flag
    from source as s
    left join current_customers as c
        on s.customer_id = c.customer_id
    where c.customer_id is null
)

```

4. Modular Query Structure – SQL scripts are designed such a way as updated_customers, expired_customers, new_customers to **reuse logic efficiently** without redundancy. This improves readability and allows Snowflake's optimizer to execute CTEs more intelligently.

```

scd2_customers - Notepad
File Edit Format View Help
c.customer_name,
c.email,
c.join_date,
c.active,
c.load_date,
false as current_flag
from current_customers as c
where exists (
    select 1
    from updated_customers as u
    where c.customer_id = u.customer_id
)
)

select * from new_customers
union all
select * from updated_customers
union all
select * from expired_customers

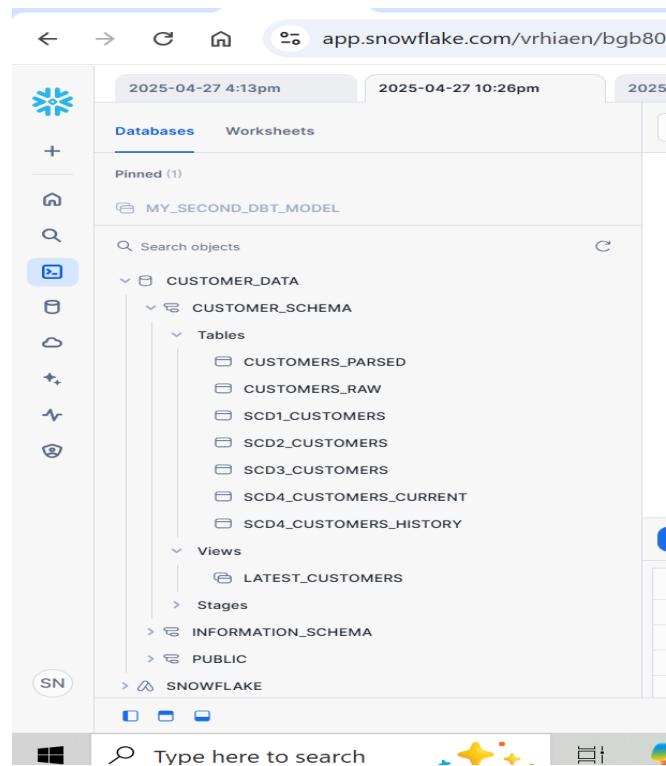
{% else %}

select
customer_id,
customer_name,
email,
join_date,
active,
current_timestamp() as load_date,
true as current_flag
from source

{% endif %}

```

5. DBT Materializations- Separate models are created such as tables, views, incremental models based on use cases and faster access. For example, if they need only latest records, they can access latest_customers view. If they need only history, they can access scd4_history table.



Code Reusability and Modularization

1. Reusable Models - Created separate DBT models for each SCD type (scd1_customers, scd2_customers, scd3_customers, scd4_customers_current, scd4_customers_history). Each model uses {{ ref() }} to reference upstream models instead of hardcoding table names, enabling easy reuse and dependency management. (Refer SCD scripts)

2. YAML-Driven Documentation and Tests – Created single .yml file to organize column level descriptions and reusable tests. Tests and docs are linked to each model dynamically, keeping code DRY (Don't Repeat Yourself).

```
schema - Notepad
File Edit Format View Help

models:
  - name: scd1_customers
    description: "SCD Type 1: overwrites existing data with the latest version"
    columns:
      - name: customer_id
        description: "Unique customer ID"
        tests:
          - not_null
          - unique

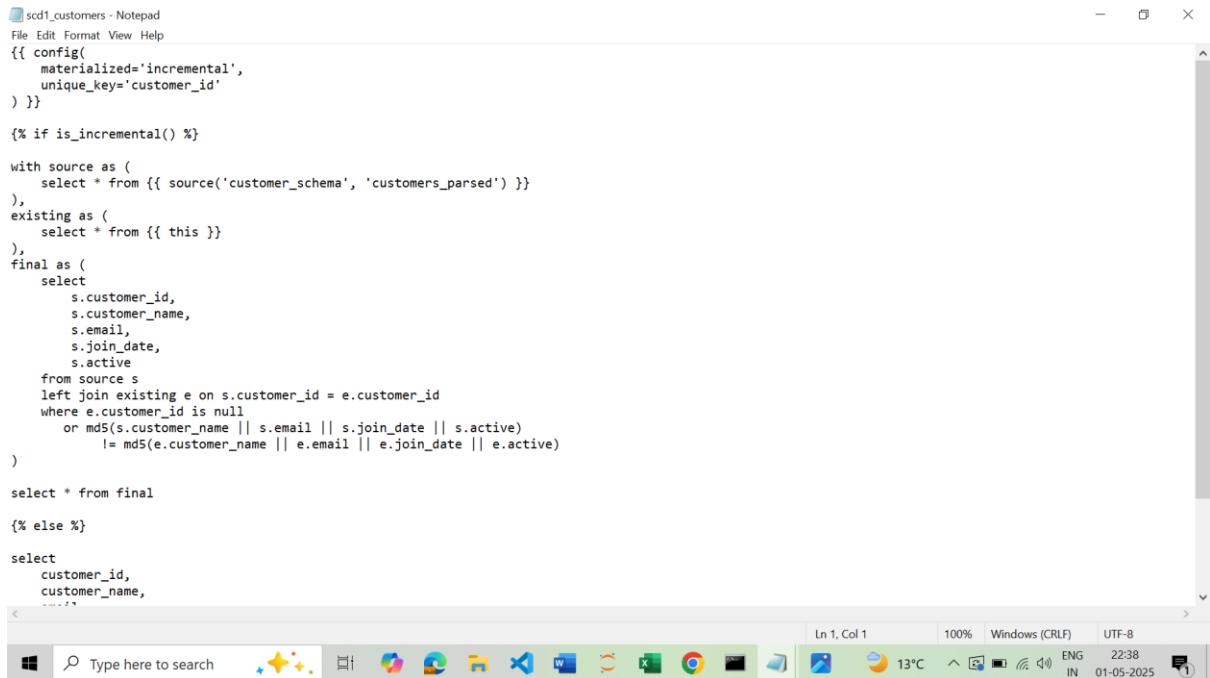
      - name: customer_name
        description: "Customer name"
        tests:
          - not_null

      - name: email
        description: "Customer email"
        tests:
          - not_null

      - name: active
        description: "Active status"
        tests:
          - accepted_values:
              values: [TRUE, FALSE]

  - name: scd2_customers
    description: "SCD Type 2: tracks historical changes with flags and dates"
    columns:
      - name: customer_id
        description: "Unique customer ID"
        tests:
          - not_null
```

3. Configurable Materialization - Leveraged DBT's {{ config() }} block to parameterize materialization (e.g., materialized='incremental'), allowing reuse of the same pattern across models without duplicating logic. This modular structure makes the pipeline scalable and maintainable—new dimensions or changes can be implemented by adding or updating small, isolated modules instead of touching everything.



```

scd1_customers - Notepad
File Edit Format View Help
{{ config(
    materialized='incremental',
    unique_key='customer_id'
) }}

{% if is_incremental() %}

with source as (
    select * from {{ source('customer_schema', 'customers_parsed') }}
),
existing as (
    select * from {{ this }}
),
final as (
    select
        s.customer_id,
        s.customer_name,
        s.email,
        s.join_date,
        s.active
    from source s
    left join existing e on s.customer_id = e.customer_id
    where e.customer_id is null
        or md5(s.customer_name) || s.email || s.join_date || s.active
        != md5(e.customer_name) || e.email || e.join_date || e.active
)
select * from final
{% else %}

select
    customer_id,
    customer_name,
    ...

```

The screenshot shows a Windows Notepad window titled "scd1_customers - Notepad". The content of the file is a Python script using the Fivetran incremental loading syntax. It defines a configuration block, checks if it's an incremental load, and then defines three parts: source, existing, and final. The final part uses a left join to update the source table with data from the existing table, filtering by customer_id. The script then selects all columns from the final table. The Notepad window has a standard Windows title bar and taskbar at the bottom.

4. Centralized Source Definition – Customers_parsed table is defined such that only once in .yml file. All other models pulls data from the same source file every time ensuring **no duplication and consistency** across all files.

5. Parameter-Driven Design- Introduced parameters such that future attributes changes can be added within same column with out duplicating another column. For example, if persons email id get changed twice, its get updated in that list rather than creating separate list.

Rubric Criteria	STATUS
Environment setup and repository configuration	DONE
Correct implementation of incremental loading -Create all models appropriately - Create mappings appropriately	DONE
Proper Error handling and Logging	DONE
Optimization techniques used	DONE
Code reusability and modularization	DONE
Documentation Quality	DONE